Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 34 (2014) 249 – 256

Procedia Computer Science

The 9th International Conference on Future Networks and Communications (FNC-2014)

A Hybrid Approach for Scheduling Virtual Machines in Private Clouds

Heba Kurdi[a]*, Ebtehal T. Alotaibi[b]

[a] *Computer Science Department, King Saud University, SA*
[b] *Computer Science Department, Al Imam Muhammad Ibn Saud Islamic University, SA*

Abstract

Quality of Service (QoS) support in private clouds is a challenging process because of the limitations of available resources and the high rate of received jobs, which leads to an NP hard scheduling problem. In private clouds, resource owners are usually interested in maximizing their resource utilization and completion rates while minimizing the turnaround time of their jobs, which complicates the scheduling problem even more. Haizea is an eminent cloud scheduler that offers high performance in terms of job turnaround time and completion rate. However, Haizea, and cloud schedulers in general, suffer from low resource utlization. Additionally, cloud schedulers usually consider only end users' demands, while providers' demands are entirely neglected. This is because an infinite pool of resources is assumed, which is difficult to achieve and simply not true in private clouds. Conversely, Condor, the eminent High Throughput Computing (HTP) scheuler, is known for addressing these shortcomings by formulating owner's and user's requirements as a logical expression evaluated based on the context which result is high resource utilization. Unfortunatly, this comes with the price of long execution time. As each of Haizea and Condor has its own advantages and limitations, in this paper, we propose a hybrid Haizea and Condor approach (HHCS) which utilizes techniques from both schedulers in a way that maximizes their advantages and overcomes their limitations. The proposed approach has been tested thoroughly in a simulated private cloud environment under various numbers of nodes and jobs. Experimental results illustrated an enhanced performance in terms of resources utilization without compromising the job turnaround time or the job completion rate.

Selection and peer-review under responsibility of Conference Program Chairs

Keywords: Haizea; Cloud computing; virtual machines; Condor; scheduling

1. Introduction

Cloud computing is defined as an Information Technology (IT) sourcing and delivery model for enabling

* Corresponding author. Tel.: +966 555669791; fax: +966 1 2581616.
E-mail address: heba.kurdi@gmail.com

Selection and peer-review under responsibility of Conference Program Chairs
doi:10.1016/j.procs.2014.07.019

convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management efforts or service provider interactions. There are four cloud delivery models: Private clouds where cloud services are provided solely for an organization and are managed by the organization or a third party; Community clouds where cloud services are shared by several organizations of common concerns; Public clouds where cloud services are available to the public and owned by a service provider; and Hybrid clouds which include combinations of the above cloud models.

Virtualization technologies are the key enabler of all cloud models by giving providers a flexible way of managing their resources. Virtual infrastructure (VI) management is a key concern when building clouds and it poses many challenges. The core of any VI management system is a VI scheduler capable of allocating resources efficiently, which is extremely challenging. In private clouds, where resources are limited, the scheduler job is even more complicated and challenging [1].

In general, scheduling is a mechanism to allocate resources to jobs with the objective to optimize one or more performance measures. Many of its forms are considered among the most difficult computational problems known as NP-hard [2]. There are already a number of cloud schedulers that allocate incoming jobs (lease requests) to virtual machines (VMs). Among the well-established VI schedulers is Haizea which implements a simple greedy allocation strategy, assuming an infinite resource capacity, as cloud schedulers usually consider public and hybrid clouds. This may result in resource under utilization problem that is critical in private clouds. On the other hand, High Throughput Computing (HTC) schedulers, such as Condor, are known for their efficient resource utilization. Condor uses a matchmaking policy, which matches a user's job to an appropriate machine based on resource owners' and users' requirements. The latter factor, addressing resource owners' requirements, is very demanding in private clouds, although not considered by most cloud schedulers. However, Condor is very slow in execution and is consequently expected to give lower data rate and job turnaround time when compared to cloud schedulers.

In this paper, we propose a hybrid approach, HHCS, that combines advantages of Haizea and Condor. Thus, the proposed scheduler augments Haizea with the Condor scheduling policy to overcome the resource utilization problem and to better addresses resource owners' requirements. To evaluate HHCS, the algorithm was implemented in Python as a pluggable policy in Haizea. The experimental results showed enhanced utilization of HHCS when compared to Haizea, without compromising other performance measures: the average job completion rate and turnaround time.

The rest of the paper is organized as follows. In section 2, Haizea and Condor schedulers are overviewed. In section 3, the proposed scheduler, HHCS, is introduced. The evaluation methodology is briefly described in section 4 while simulation results are presented and discussed in section 5. The paper is summarized and concluded in section 6.

2. Background

This work is solely dependent on the two schedulers Haizea and Condor, therefore in this section we overview both schedulers to provide a clear background for the HHCS.

Haizea [3] is an open source VM scheduler developed by the University of Chicago in Python. Haizea takes job requirements as described by the user, and makes scheduling decisions accordingly. The fundamental resource provisioning abstraction in Haizea is the lease. A lease is a form of contract where one party agrees to provide a set of resources to another party. When a user needs computational resources from Haizea, he does so in the form of a lease. The lease terms must include three dimensions: the hardware resources (CPU, memory, etc.) required by the resource consumer, the software environments that must be installed in those resources, and the period during which the hardware and software resources must be available.

The current version of Haizea supports three types of leases: (1) Immediate (2) Best Effort (BE) and (3) Advance Reservation (AR) leases. Immediate leases are useful when a request needs immediate attention, so resources are provided instantly if available. Otherwise, the lease request is rejected. In situations where a resource request does not require immediate attention and can wait for availability of resources, BE leases are suitable. BE leases are queued up and processed in a First-In First-Out manner on availability of resources. AR lease requests can be used when users need infrastructure resources at specific start and end times. Immediate and AR leases can cause an already running BE lease to be suspended whenever they require resources [4].

Scheduling jobs in Haizea, is carried out by reading accepted leases periodically and mapping each lease to suitable virtual machines based on the lease type; (1) If the accepted lease is an AR or immediate lease, Haiza will instantly check the availability of its required resources to grant or deny the lease. If the required resources are free or busy

with BE leases, the AR/immediate lease will be scheduled on those resources based on the requested time, and the running BE lease (if any) will be preempted, suspended and queued, (2) If the accepted lease is a BE lease, Haizwa will queue the lease, all jobs in the queue will be scheduled in First-Come-First-Serve queue with backfilling. When the start time of a scheduled lease is due the leases id de-queued, and a host will selected for the lease based on the host score which is calculated according to three factors: (a) nodes with fewer leases to preempt, (b) nodes with the highest capacity (c) nodes where the current capacity doesn't change for the longest time. The weighted sum of those factors represents the host score. The output of the scheduling process will be a schedule containing available resources assigned to the jobs on each timeslot.

Condor [1] is an open source batch system for high throughput computing developed by Wisconsin University in C language. Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring and resource management [1], [2]. It uses a matchmaker to map the user's jobs to appropriate machines. Both jobs and machines are described by the ClassAds (short for classified advertisements) language. ClassAds provide schema-free descriptions of jobs and resources that are easy to use effectively [5],[6]. All machines in Condor pool advertise their attributes, such as available memory, CPU and speed, current load, along with other static and dynamic properties. ClassAds also advertise the preference (rank) for running jobs submitted by the users. Likewise, when submitting a job, users specify a job ClassAd with their requirements and preferences, including the type of machine (rank) they wish to use. Condor plays the role of matchmaker by continuously reading the entire job ClassAd and all the machine ClassAds, matching and ranking them making sure that all requirements in both ClassAds are satisfied [1],[2],[5],[7].

When a user submits a job, the job is immediately stored in a persistent queue, Condor matchmaker reads all queued jobs periodically and schedules them based on the degree of satisfaction of both machine and job requirements. After finding a matching machine to a particular job, the matchmaker evaluates several expressions: (1) How does the job rank the machine? Job ClassAds specify a rank expression that, when evaluated, can differentiate between different machines. (2) Is there already a job running on the machine? If so, the matchmaker evaluates the machine rank of the job and compares this rank to the machine rank of the currently running job. If the new rank is higher, the matchmaker may consider preempting the job. (3) If there is a job running on the machine, but it does not have a higher rank than the currently running job, is the owner of the job looking for a match that has a higher user priority than the owner of the currently running job? If so, the matchmaker checks if the administrator's requirements allow preemption [4].

3. Hybrid Haizea-Condor Scheduler (HHCS)

As described eailer, Haizea is an eminent cloud scheduler that offers high performance in terms of job turnaround time and job completion rate. However, Haizea, and cloud schedulers in general, suffer from low resource utlization. Additionally, cloud schedulers usually consider only end users' demands, while providers' demands are entirely neglected. This is because an infinite pool of resources is assumed, which is difficult to achieve and simply not true in private clouds. Conversely, Condor, the eminent HTC scheuler, is known for addressing these shortcomings by formulating owner's and user's requirements as a logical expression evaluated based on the context which gives high resource utilization. Unfortunatly, this comes with the price of long execution time.

Threfore, the end objective of this paper is to ensure better resource utilzation in private clouds by combining the two well known schedulers, Haizea and Conodr, to maximize their advantages and overcome their limitations. To do this, we adapted the Condor matchmaking policy to suit Haizea and implemented it along with additional functions to deal with Condor attributes for jobs and resources so that the new implentation of Haizea can address the additional requirements of owners and users. Additionally, we adapted the Condor matchmaking policy and introduced the additional jobs and resources attributes used by Condor to address resource owner and user requirements.This involoved applying the lease concept and types to the ClassAd of condor and implementing required interations with the lease manager and mapper, as well as insuring compatible interface with other Haizea componets, as shown in Figure 1. This algorithm has been implemented in Python using Haizea 1.0 as a platform.

Algorithm 1 Hybrid Haizea Condor Scheduler (HHCS)

```
Input: Leases
Output: Schedule of resources and jobs
Description:
for every period or change point
    for each received lease
        if (lease.type is AR/immediate lease)
            search for required resources
            if (there are enough resources)
                if (the resources are running BE lease)
                    preempt the BE lease;
                    suspend BE lease;
                    queue BE lease ;
                schedule the AR/immediate lease on those resources;
            else
                reject the AR/immediate lease ;
        if (lease.type is BE lease)
            enqueue (lease);
    for each l ∈ LeaseQueue
        dequeue (l);
        for each m ∈ MachinePool
            if (m.ClassAd.Type == j.ClassAd.Requirements)
                machine_score+ = 0.5;
                Matched_list[]+ = m;
            else
                machine_score = 0;
        for each (m ∈ Matched_list[])
            m = MAX (Matched_list[], j.ClassAd.Rank);
            if (there is a job p running on m)
                if (j.ClassAd.Type == m.ClassAd.Rank
                and p.ClassAd.Type != m.ClassAd.Rank )
                    preempt p;
                    queue p;
                else
                    remove m from Matched_list[];
                        m = nil;
            if (m != nill)
                schedule m for j ;
                break ;
```

Figure .1. Hybrid Haizea-Condor Scheduler (HHCS)

4. Evaluation Methodology

To evaluate HHCS, we followed a strictly controlled experimental framework on a similar approach suggested by [8]. Haizea has been used as both a benchmark and a simulation environment. Haizea provides a simulation mode to simulate cloud performance allowing any scheduling policy to be plugged easily and replace the original Haizea policy. In this mode, Haizea takes (a) a workload: a sequence of job requests and (b) a description of the simulated cluster and a (c) configuration file specifying simulation and scheduling options (such as the characteristics of the hardware to simulate), and processes them in a simulated time. The result of the simulation is a data file with raw scheduling data and metrics, which can be used to generate reports and graphs [3].

The simulated private cloud cluster consisted of a number of nodes from the values {50 , 100 , 150, 200, 250}, each node with a number of processors (CPU) selected randomly from the range [1-10], and memory amount drawn randomly from {1024, 2048, 4096, 8192}. The machine ranks and types were generated randomly from the range [1-10]. The workload was generated using the trace file from the Grid Workload Archive [9]. As required by Condor, all lease requests were considered BE jobs. The considered performance measures were:

- γ Average hardware utilization: The CPU utilization is one of the ready metrics provided by Haizea simulator. It is calculated as:
  $\gamma = \frac{\sum_{i=0}^{m}(hardware\ utilization)}{m}$, where m is the total number of occupied machines. (1)
- α average turnaround time of the jobs (Seconds): The average job turnaround time is not included among the Haizea probes and therefore we coded it as the summation of job waiting times and job run times divided be

the number of utilized machines [10]:

$$\alpha = \frac{\sum_{i=0}^{n}(wait\ time(i) + run\ time(i))}{n}$$, where n is the total number of jobs. (2)

- δ job completion rate: The job completion rate is calculated as the ratio of completed jobs to requested jobs at each time step:

$$\delta = \frac{number\ of\ completed\ jobs}{number\ of\ requested\ jobs}$$ (3)

5. Results and Discussions

The end objective was to answer the question: is a hybrid algorithm that combines features of Condor and Haizea able to provide higher resource utilization for private clouds? To answer this question, we employed the Haizea simulator to develop a private cloud model in a Dell M5110 laptop, Intel(R) Core(TM) i7 CPU@2.20GHz and 8.00 GB for RAM 64-bit Ubuntu 12.10 Operating System. A series of experiments were conducted to compare between Haizea and HHCS under different settings. Experiments were divided into two sets: in the first set, we used a fixed number of nodes while varying the inter-arrival time of lease requests and in the second set, the numbers of nodes varied while the inter-arrival time was fixed. Results were collected based on three performance measures: the average turnaround time, hardware utilization and job completion rate.

5.1. Fixed Nodes Variable Inter-arrival Time

This set of experiments aimed at studying how hardware utilization, job turnaround time and completion rate would be affected by the amount of load in the system, based on varying the inter-arrival time of lease requests, when HHCS or Haizea are used for scheduling lease requests. The number of required nodes was fixed on 50 then 250 nodes and the inter-arrival time was variable, with the values {32, 64, 128, 256, 512, 1024, 2048. The result of these 14 experiments is summarized in Figure 2-Figure 4.

- CPU Utilization

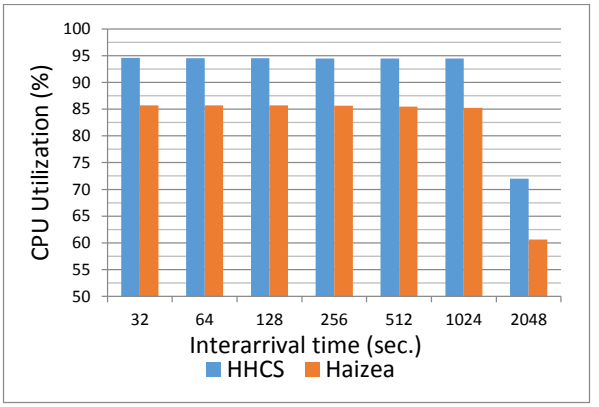
Figure 2 (a) and Figure 2 (b) illustrate both scheduler behaviors when the number of required nodes was fixed at 50 and 250 respectively. The results show the superiority of HHCS performance in terms of utilization, where it maintained much higher utilization in all scenarios especially at a large number of nodes, where Figure 2 (b) shows that the difference between the utilization of the schedulers can reach up to 10%. Additionally, Figure 2 shows that, as expected, increasing the inter-arrival time for both Haizea and HHCS would decrease resource utilization, as less requests would arrive to the cloud. When the number of requested nodes increased from 50 to 250 nodes, the difference in utilization between the two schedulers was magnified in favor of HHCS.

- Turnaround time

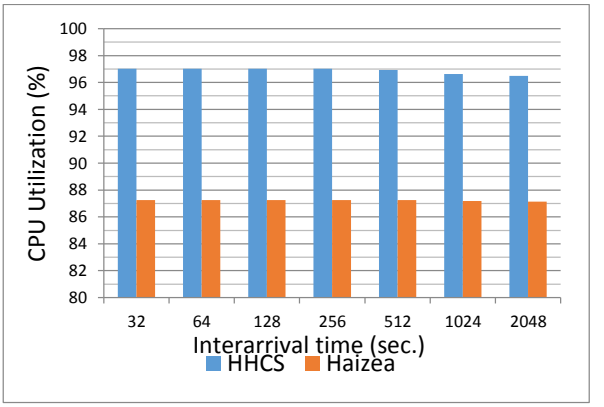
Figure 3 (a) and Figure 3 (b) show the achieved turnaround time, by HHCS and Haizea, when the number of required nodes was fixed at 50 and 250 respectively. In all cases, Haizea and HHCS produce close performance. As anticipated, Figure 3, shows that increasing the inter-arrival time would decrease the job turnaround time, while increasing the number of requested nodes would have the opposite effect. This is because the system would be less loaded.

- Job Completion rate

Figure 4 (a) and Figure 4 (b) illustrate the job completion rate of both schedulers when the number of requested nodes was fixed at 50 and 250 respectively. In all cases, HHCS has achieved nearly an identical performance to Haizea. However, Haizea showed slightly higher job completion rate for a lower number of requested nodes, as illustrated in Figure 4 (a), while the HHCS has an improved performance when the number of requested nodes is higher at 250 nodes as shown in Figure 4 (b). The results in Figure 4 emphasize that increasing the inter-arrival time for both Haizea and HHCS or decreasing the amount of requested nodes would be reflected positively on the job completion rate as the system would be able to process more lease requests.

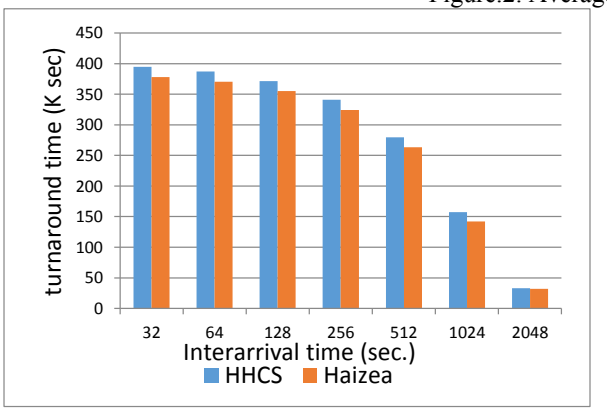


(a) 50 nodes

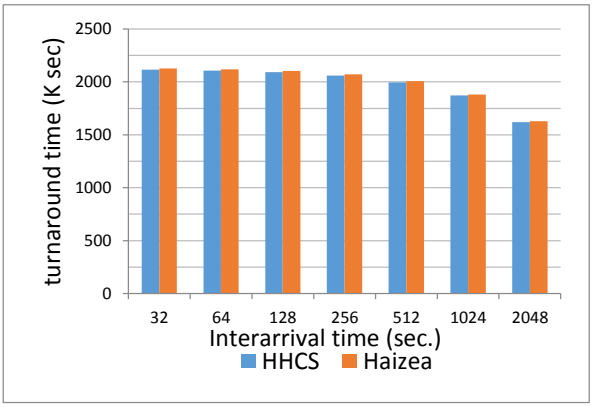


(b) 250 nodes

Figure.2. Average CPU utilization

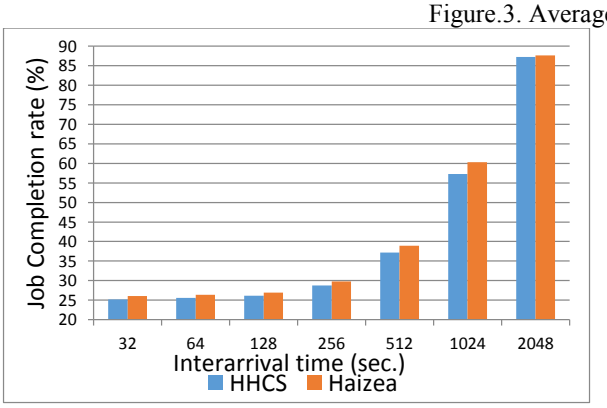


(a) 50 nodes

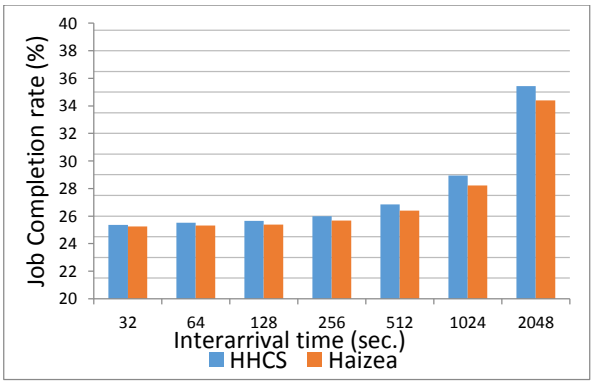


(b) 250 nodes

Figure.3. Average turnaround time



(a) 50 nodes

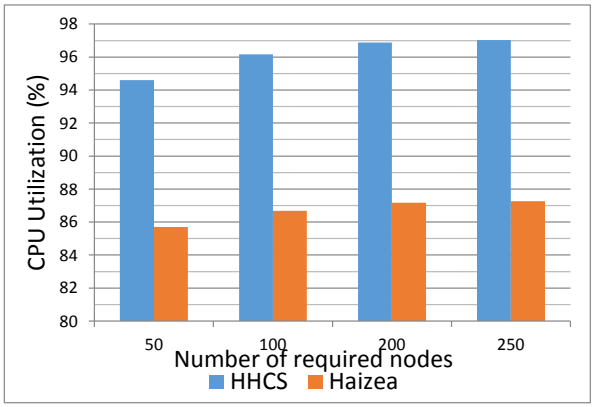


(b) 250 nodes

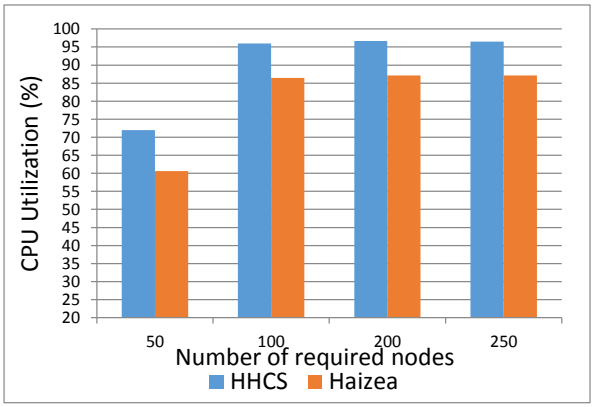
Figure.4. Average jobs completion rate

5.2. Variable Node Fixed Interarrival Time

In this set of experiments, we varied the number of required nodes using the values {20, 100, 150, 200, 250}, while considering two cases for inter-arrival time: 32 and 2048. The aim was to study how systems turnaround time, hardware utilization and job completion rate would be affected by the amount of load in the system, based on varying the number of requested nodes when HHCS or Haizea are used for scheduling lease requests. The results of these 10 experiments are summarized in Figure 5-Figure 7.

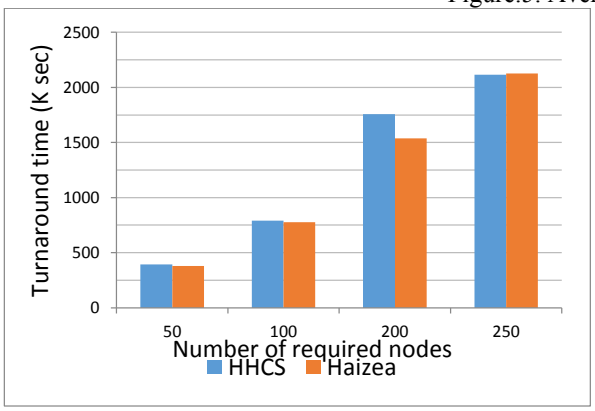


(a)interarrival time 32 seconds

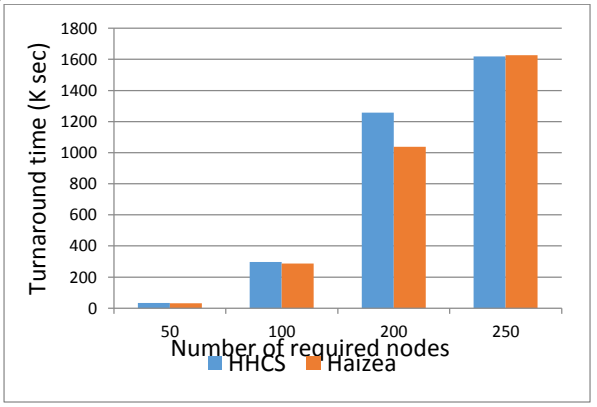


(b) interarrival time 2048 seconds

Figure.5. Average CPU utilization

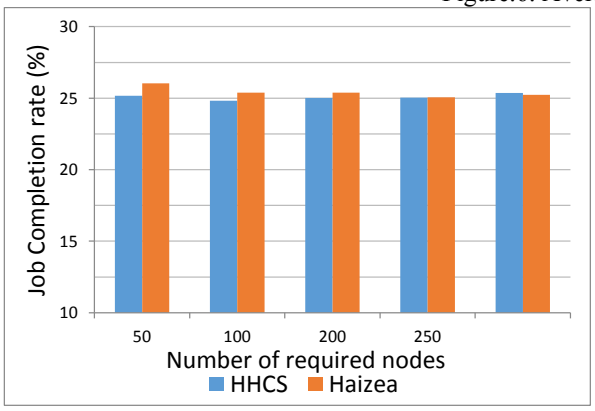


(a) interarrival time 32 seconds

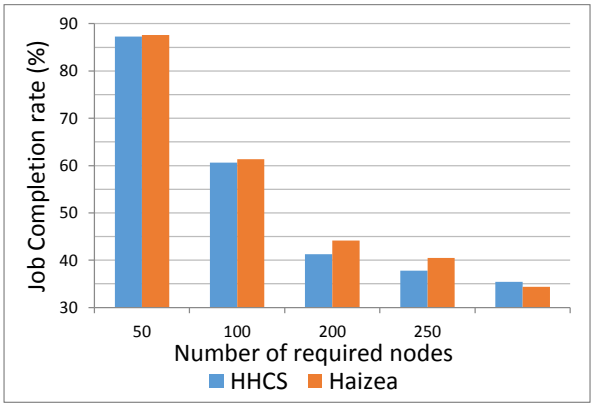


(b) interarrival time 2048 seconds

Figure.6. Averag e turnaround time



(a) interarrival time 32 seconds



(b) interarrival time 2048 seconds

Figure.7. Average jobs completion rate

- CPU Utilization

Figure 5 (a) and Figure 5 (b) illustrate both schedulers behaviors when the inter-arrival time was fixed at 32 and 2048 respectively. The results show that in all cases, HHCS resulted in higher resource utilization compared to Haizea, regardless of the inter-arrival time, with difference of at least 10%. Additionally, Figure 5 shows that, as expected,

increasing the job inter-arrival time, from 32 to 2048, would decrease the resources utilization in both Haizea and HHCS as less requests would arrive to the system.

- Turnaround time

Figure 6 (a) and Figure 6 (b) illustrate the average turnaround time obtained by HHCS and Haizea, when the inter-arrival time was fixed at 32 and 2048. In most cases, HHCS and Haizea have a similar performance. As predicted, increasing the number of requested nodes enlarged the job turnaround time. On the other hand, increasing the inter-arrival time decreased the turnaround time. This is because the cloud infrastructure would be less loaded.

- Jab completion rate

Figure 7 (a) and Figure 7 (b) illustrate the job completion rate of both schedulers when the inter-arrival time was fixed at 32 and 2048. In all cases, HHCS has achieved nearly similar performance to Haizea. The results in Figure 7 suggest that increasing the number of requested nodes would have marginal effects on job completion rate for both Haizea and HHCS, especially for short inter-arrival times. However, increasing the job inter-arrival time would result in better job completion rate and increase the difference in performance between the two schedulers.

6. Conclusion

The core of any cloud is a scheduler capable of efficiently allocating incoming job requests to virtual resources, which is extremely challenging. In private clouds, where resources are limited, the scheduler job is even more complex. In general, cloud schedulers, such as Haizea, assume infinite resource capacity when allocating clients' requests to their virtual resources. Although this is true for public and hybrid clouds, it is simply not true and results in low resource utilization in private clouds. Conversely, Condor, the eminent High Throughput Computing scheduler is known for its high resource utilization. Unfortunately, this comes at the price of long execution time.

Therefore, in this paper we proposed a hybrid scheduler (HHCS) that combines the two well-known schedulers, Haizea and Conodr, to maximize their advantages and overcome their limitations with an objective to ensure better resource utilization in private clouds. A series of experiments were conducted to evaluate HHCS under different settings. Results illustrated an enhanced resource utilization of the proposed scheduler, when compared to Haizea, without compromising the turnaround time or the job completion rate. As a future work, we are preparing to evaluate HHCS on a real test bed that has been constructed specifically for this purpose. Additionally, a detailed comparative study is intended to compare between Condor and HHCS.

Acknowledgements

This work was funded by the Long-Term Comprehensive National Plan for Science, Technology and Innovation of the Kingdom of Saudi Arabia, grant number 11- INF1895-08.

References

1. T.Tannenbaum, M. Livny D.Thain, "Condor and the Grid," in Grid Computing: Making the Global Infrastructure a Reality., 2003.
2. HTCondor™ Version 8.0.0 Manual. Madison: Center for High Throughput Computing, University of Wisconsin, 2013.
3. B. Sotomayor. (2009, February 16) www.cloudbook.net. [Online]. http://www.cloudbook.net/community/contributors/borja-sotomayor
4. Chokhani, P.; Somani, G., "Dynamic resource allocation using auto-negotiation in Haizea," Contemporary Computing (IC3), 2013 Sixth International Conference on , vol., no., pp.232,238, 8-10 Aug. 2013
5. M.Livny A.Roy, "CONDOR AND PREEMPTIVE RESUME SCHEDULING," in Grid Resource Management: State of the Art and Future Trends. Madison: kluwer academic, 2004.
6. D.wright,K.miller,M.livny T.tannenbaum, "Condor: A Distributed Job Scheduler," in Beowulf Cluster Computing with Linux, 2001, p. 44.
7. T. Tannenbaum, and M. Livny D. Thain, "Distributed computing in practice: the Condor experience," vol. 17, no. Issue 2-4, pp. 323 - 356 , February 2005.
8. M Silva et al., "CloudBench: Experiment Automation for Cloud Environments," in IEEE International Conference on Cloud Engineering (IC2E), 2013, pp. 302-3011.
9. DS Group. (2007, desu) The Grid Workloads Archive. [Online]. http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-3
10. M. . Saj id and Z. Raza, "Level Based Tasl< Duplication Strategy to Minimize the Job Turnaround Time," in 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012.